VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Strategická hra pro Android

Strategic Game for Android

2013 Peter Gorčák

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Peter Gorčák**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: Strategická hra pro Android
Strategic Game for Android

Zásady pro vypracování:

Cílem práce je vytvořit strategickou hru pro mobilní platformu Android. Hra bude kombinovat prvky tahové strategie a real-time soubojů. Protihráčem bude umělá inteligence, která bude implementována v rámci této práce. Dění bude probíhat v animovaném herním světě zobrazovaném pomocí izometrického zobrazení.

1. Návrh tématu hry, pravidel a ovládání.
2. Návrh vizuální části hry, tvorba 3D prostředí (izometrická grafika, animace postav)
3. Zpracování a vykreslování hry pomocí OpenGL/Canvasu.
4. Vykreslování scény, textur a animace pomocí API rozhraní OpenGL/Canvasu.
5. Implementace AI.

Seznam doporučené odborné literatury:

[1] Cay S. Horstmann, Core Java(TM), Volume I--Fundamentals, Prentice Hall, 2007, ISBN-13: 978-0132354769
[2] Reto Meier, Professional Android 2 Application Development, Wrox, 2010, ISBN-13: 978-0470565520
[3] Sayed Hashimi, Pro Android 2, Apress, 2010, ISBN-13: 978-1430226598

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Mgr.Ing. Michal Krumnikl**

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2013

prof. RNDr. Vladimír Vašinek, CSc.
vedoucí katedry

VYSOKÁ ŠKOLA BÁŇSKÁ - TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave 5. 5. 2013 .....

Na tomto mieste by som rád poďakoval vedúcemu práce pánovi Mgr. Ing. Michalovi Krumniklovi za cenné rady a pripomienky, rodine a priateľom za morálnú podporu počas tvorby tejto práce.

Abstrakt

Táto bakalárská práca sa zaoberá tvorbou a implementáciou ťahovej strategickej hry určenej pre operačný systém Android. Hra podporuje minimálnú verziu systému od 3.0 Honeycomb alebo vyššie. Ladená bola na verzii 4.0.4 Ice Cream Sandwich. Hra je rozdelená na dve časti - ťahová globálná mapa a real-time bitky. V práci sú popísané princípy a pravidlá samotnej hry, jej implementácia spolu s Frameworkami ktoré sú k dispozícií a aj technické problémy s ktorými som sa počas tvorby stretol. Záver práce popisuje tvorbu jednoduchej umelej inteligencie. Hra je vytvorená pomocou programovacieho jazyka Java vo vývojovom prostredí Eclipse ktoré je oficiálne podporované spoločnosťou Google pre tvorbu Android aplikácií.

Kľúčové slová: Android, cyklus, framework, hardwarová akcelerácia, hra, metóda, trieda, vlákno

Abstract

This bachelor thesis deals with the creation and implementation of the turn based strategic game designed for the operating system Android. This game requires minimum version of the system 3.0 Honeycomb or higher and has been debugged on version 4.0.4 Ice Cream Sandwich. It is splitted into two parts - the turn based global map and the real-time battles. This thesis describes also the principles and the rules of the game itself, its implementation together with the Frameworks that are available and also technical problems with which i met during production. The final part of the thesis describes the creation of a simple Artificial Inteligence. The game is developed using programing language Java in Eclipse development environment that is officially supported by Google for creating Android aplications.

Keywords: Android, loop, Framework, hardware acceleration, game, method, class, thread

Zoznam použitých skratiek a symbolov

AI	–	Artificial Intelligence
API	–	Application programming interface
MB	–	Gigabyte
GPU	–	Graphic processing unit
MB	–	Megabyte
RAM	–	Random Access Memory
XML	–	eXtensible Markup Language

Obsah

Zoznam obrázkov

Zoznam výpisov zdrojového kódu

1 Úvod

Operačný systém Android je open source platforma používaná prevažne na mobilných zariadeniach. Systém je založený na linuxovom jadre verzie 2.6 a bol navrhnutý aby umožňoval beh na rozdielnom hardwari. Verejne sa zviditeľnil v roku 2005 keď bol startup projekt Android Inc. odkúpený spoločnosťou Google a v roku 2008 vydaním verzie 1.0 vstúpil na trh s mobilnými operačnými systémami [1, 2].

Počas ďalších rokoch vývoja sa stal Android veľmi populárným a dnes je jedným z najpoužívanejších mobilných operačných systémov. Android ponúka vývojárom širokú škálu možností a podpory pri tvorbe vlastných aplikácií. K ním patria aj hry a samotný herný priemysel v súčastnosti nie je vôbec zanedbateľný a rozvíja sa aj na mobilných zariadeniach používajúcich systém Android. Tieto hry sú čím ďalej tým náročnejšie a kvalitnejšie a raritou už nie sú ani prepracované 3D hry v deatilnom grafickom prevedení.

V tejto práci popisujem vývoj ťahovej strategickej hry pre OS Android. Táto hra sa skladá z dvoch rozdielných častí. Prvá je v 2D grafickom prevedení a odohráva sa na globálnej mape. Druhá časť hry sú real-time bitky s izometrickou grafikou. V kapitole 2 je popísaná samotná hra, jej pravidlá a princípy podľa ktorých funguje. Kapitola 3 rozoberá a približuje detalnejšie dostupné technológie a frameworky/enginy určené na tvorbu hier pre OS Android. V kapitole 4 je detailne popísaná implementácia a tvorba samotnej hry spolu s tvorbou grafickej a zvukovej stránky hry. Kapitola 5 vysvetľuje návrh a vytvorenie jednoduchej AI použitej v hre. Záverečná kapitola 6 sa venuje problémom ktoré sa počas tvorby hry vyskytli a ich riešeniam. Záver kapitoly približuje ladenie hry a zariadenia ktoré boli pri tom použité.

2 Popis hry, pravidiel a jej princípov

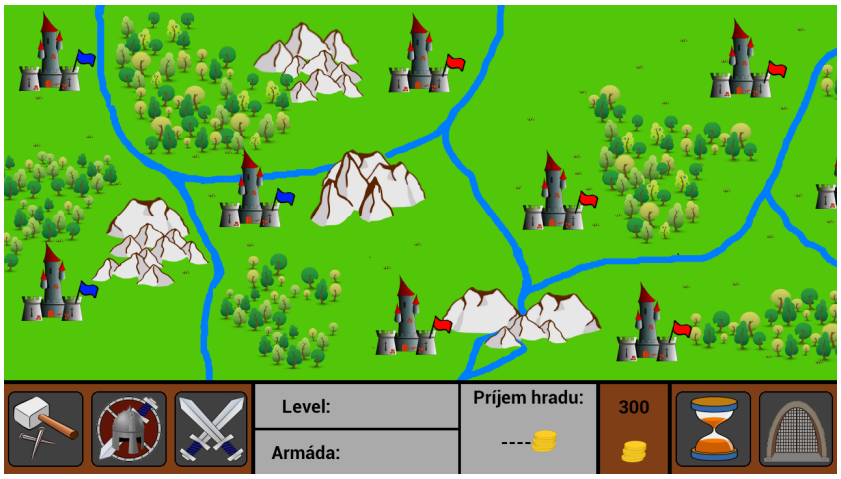
Táto kapitola popisuje celú hru, jej rozdelenie, základné princípy a vlastnosti každej časti a druhy vojenských jednotiek ktoré sa v hre vyskytujú. Záver kapitoly je venovaný porovnaniu už s podobnými existujúcimi hrami dostupnými na Googpe Play.

Hra je ťahová stratégia s real-time bitkami ktoré sú rozdelené na menšie útoky. Cieľom hry je aby hráč obsadil celé územie ktoré je rozdelené na menšie časti zobrazené na ťahovej mape. V tom sa mu snaží zabrániť AI. Ak hráč obsadí všetky územia, vyhral celú hru, ak naopak, obsadí všetky územia AI a hráčovi neostane žiadny hrad z ktorého by mohl znovu útočiť, hru prehral. Postupom hry hráč vylepšuje tieto hrady aby mohol zarábať viac zlata, prípadne odomykať viaceré vojenské jednotky s ktorými tieto územia dobýja. Rovnako ako hráč aj AI postupom hry vylepšuje svoje hrady a dokupuje ďalšie vojenské jednotky. Pri štarte novej hry začína hráč iba s pár hradmi a väčšinu územia mapy ovláda AI. Samotná hra je do dvoch rozdielných častí:

- Ťahová časť odohrávajúca sa na globálnej mape
- Real-time bitky

2.1 Ťahová časť

V tejto časti hry sa nachádza globálna mapa ktorá je rozdelená na šestnásť území a každé toto územie je kontrolované jedným hradom. Hrady sú od seba rozoznateľné malými vlajkami ktoré značia jeho majteľa. Každý takýto hrad má svoju úroveň ekonomiky, armády a zoznam susedných hradov na ktoré je možné zaútočiť ak sú pod kontrolou oponenta. Ekonomická a armádna úroveň určujú finančný prínos daného hradu na konci ťahu a maximálné množstvo vojenských jendotiek a rovnako aj ich typ ktoré môžu byť v hrade držané a kupované. Na obrázku 2.1 je ukážka z hry v tejto časti.



Obr. 2.1: Ukážka časti hry na mape

Aby hra nebola príliš jednoduchá a krátka, hráč alebo AI môžu v jednom ťahu vykonať v každom meste len jednú z troch možností:

- Výstavba
- Nábor
- Útok

2.1.1 Výstavba

Hráč ma možnosť vybrať jednu z dvoch možností:

- Vylepšenie ekonomickej úrovne hradu
- Vylepšenie vojenskej úrovne

Vylepšenie ekonomickej úrovne má za následok vyšší príjem hradu po ukončení ťahu. Toto sa odrazí na celkovom množstve finančných prostriedkov ktorými hráč počas hry disponuje. Každá vyššia úroveň poskytuje väčší príjem.

Vylepšenie vojenskej úrovne zvyšuje maximálnú kapacitu vojska držaného v hrade a tým aj jeho väčšiu obrany schopnosť prípadne šancu vyhrať útok na iný hrad. Každá vyššia úroveň dovoľuje nábor drahších a tým aj kvalitnejších vojenských jednotiek.

Obe tieto úrovne sú s rastúcim levelom drahšie.

2.1.2 Nábor

Hráč má možnosť kúpiť do hradu nového vojaka ak to dovoľuje kapacita. S rastúcou vojenskou úrovňou sa odomykajú drahšie a odolnejšie typy vojakov. Každý vojak má rozdielné vlastnosi v cene, počte životov, obrane, útoku.

Maximálný počet vojakov v každom hrade je šesť a za každého vojaka na konci ťahu platí hráč rovnako AI určitý žold. V hre sú k dispozícií tieto typy vojakov:

- Kopijník
- Halapartník
- Peší rytier
- Rytier na koni

2.1.2.1 Kopijník Základná vojenská jednotka. Ide o najlacnejšieho vojaka v hre, má najmenší počet životov ako aj nízku obranu a útok. Jediná jeho výhoda je bonus pri útoku v boji proti Rytierovi na koni, vtedy má jeho útok trojnásobnú silu.

2.1.2.2 Halapartník Ide účinnejšiu verziu kopijníka. Má vyšší počet životov a obranu, útok ostáva rovnaký aj s bonusom pri boji s Rytierom na koni.

2.1.2.3 Peší rytier Najodolnejšia pešia jednotka v hre. Má najvyšší počet životov, útok a aj obranu. Spomedzi peších vojenských jednotiek je ale najdrahší.

2.1.2.4 Rytier na koni Spomedzi všetkých vojenských jednotiek je najsilnejší ale aj najdrahší. Má vysoký počet životov, úroveň obrany aj útoku. Jeho jediná slabina je v boji proti kopijníkovi alebo halapartníkovi ktorí maju pri boji s jazdcom bonus pri útoku.

2.1.3 Útok

Pri tejto možnosti sa okolo aktuálne zvoleného hradu zvýraznia nepriateľské hrady pomocou červeného obdĺžnika a hráč môže na jeden z takto označených hradov zaútočiť. Pri spustení útoku sa hra na globálnej mape pozastaví a prepne sa do režimu bitky.

2.2 Real-time bitka

V tejto časti hry sa odohráva najdôležitejšia časť celej hry - bitka o hrad. Herné pole je rozdelené na tri vodorovné línie zoradené pod sebou a hráč spolu s AI musia na každú z týchto línií umiestniť jedného vojaka ktorý je postavený proti nepriateľskému. Umiestnenie ďalšieho vojaka na líniu je možné až keď pôvodný vojak padol. Ak má hráč alebo AI nedostatok vojakov aby obsadil každú líniu, voľná línia automaticky pripadá nepriateľovi, je označená vlajkou a už sa na nej nebojuje. Pre výhru v bitke a obsadenie mesta, prípadne jeho ubránenie je potrebné obsadiť minimálne dve tieto línie.

Pri úspešnom útoku a vyhratí bitky sa zvyšok útočiaceho vojska presunie do obsadeného hradu a vojsko obrancov je úplne zničené. S dobytým hradom v aktuálnom ťahu už nie je možné s hradom vykonať žiadnu z činností. Pri neúspešnom útoku na hrad je vojsko útočníka vrátené späť do hradu odkiaľ útočilo a zvyšok armády obrancov nie je zničený a mesto ostáva pod jeho správou. Na obrázku 2.2 je ukážka bitky z hry.

Obr. 2.2: Ukážka časti hry počas bitky

2.3 Porovnanie s inými hrami

V tejto kapitole sa nachádza stručné porovnanie hry ktorej tvorba je opísaná v tejto práci a rozdiely s niekoľkými podobnými hrami dostupnými na Google Play. Strategicky zameraných hier s možnosťami ťahov pre Android nie je mnoho, často ide o hry typu Tower Defense, preto som vybral na porovnanie hry ktoré obsahujú aspoň niekoľko princípov a možností podobných s aktuálne opisovanou hrou.

2.3.1 Popis hier

Všetky hry zvolené na porovnanie sú v určitom smere podobné alebo rozdielné navzájom a aj s hrou vytvorenou počas tvorby tejto práce.

2.3.1.1 Empire versus Orcs Fantasy stratégia ktorá obsahuje iba realtime bitky vo veľmi peknej a prepracovanej izometrickej grafike. Hráč počas behu hry pridáva na bojisko rôzne druhy vojenských jednotiek ktoré si odomkol dosiahnutím určitého levelu. Cieľom hry je dostať sa na druhú stranu bojiska, zničiť obranu oponenta a popri tom chrániť takzvané bane z ktorých hráč ťaží suroviny na nákup vojska.Hra je pomerne jednoduchá, po odohraní bitky sa hráčovi pričíta skóre a nasleduje ďalšia bitka.

2.3.1.2 Age of Conquest 2D stratégia rozdelená na ťahy odohrávajúca sa na mape sveta. Hráč obsadzuje susedné územia a na každom z nich má možnosť vykonať niekoľko z piatich ožností. Po dlhšom hraní je hra stereotypná ale ostáva naďalej docela náročná. Nevýhodou je absencia real-time bitiek o územia.

2.3.1.3 Spartan Wars Budovateľská stratégia v izometrickej grafike. V hre je mnoho možností pri budovaní a rozvoji vlastného mesta, jeho úprav a miestami to pôsobí celá hra zmätene a neprehľadne. Hra je sústredená na budovanie a tak sú súboje iba v podobe kartovej hry ktorá je celkom jednoduchá.

2.3.1.4 Kingdom Battle Ide o jednoduchú hru ktorá je tiež v izometrickej grafike.V hre nie je možnosť budovania alebo kupovania vojenských jednotiek. Celá hra prebieha na mape kde sa nachádza niekoľko hradov ktoré sú medzi sebou prepojené cestami. Hráč posiela zo svojich hradov na jednotlivé cesty vojakov ktorí putujú do susedných a snažia sa ich obsadiť. V pôvodnom hrade sa medzitým doplňuje ich počet. V hre sa nenachádzajú žiadne bitky, ak sa stretnú proti sebe vojaci, jednoducho obaja zmiznú, pretože sú si rovnocenní. Vyhráva teda ten kto pošle na cestu do hradu viac vojakov. Ovládanie hry je jednoduché a intuitívne a rovnako dostatočná je aj jej grafická stránka.

2.3.1.5 Defense Craft Strategy Táto hra je v podstate Tower Defense ktorá obsahuje prvky real-time stretégií. Hráč bráni svoj hlavný hrad pomocou malých vežičiek a má možnosť aj verbovať vojakov ktorých môže posielať kdekoľvek na mape. Počas hry je možné budovy aj vylepšovať čo prináša hráčovi viac možnosti v obrane.

2.3.2 Rozdiely

V hre bolo cieľom niektoré časti zmeniť alebo zjednodušiť. Hlavným cieľom zjednodušenia hry boli možnosti a ovládanie počas jej priebehu. Ovládanie pomocou dotykov na tablete alebo mobilnom telefóne nie je tak pohodlné ako na počítači a priveľa ovládacích prvkov pôsobí nepohodlne, čo vidieť na úspechu prevažne jednoduchých hier ako napríklad Angry Birds. Preto hra obsahuje iba jednoduché možnosti a ovládacie prvky počas hrania ktoré sú pre tento typ hry potrebné.

Ďalším rozdielom je prevedenie ťahov v hre. Vďaka tomu že sú obmedzené, hráč počas hry musí zvážiť svoj postup a rozhodnúť sa ktorú činnosť vykoná, čo robí hru náročnú aj v neskorších fázach keď už má odomknuté všetky vylepšenia a vojenské jednotky.

Odlišnou časťou hry od porovnávaných je aj real-time bitka. Priebeh bitky nie je veľmi predvídateľný len na základe typu vojakov ktorí sa v bitke nachádzajú. Vojaci sú rozdelení do skupín podľa kvality, teda od najslabších po najsilnejších a majú rozdielné vlastnosti. V hre však môže nastať situácia, kedy sú proti sebe postavení dvaja rozdielní, ktorí predtým v bitke nebojovali a slabší z nich porazí silnejšieho. Prítomnosť vyššieho levelu vojakov teda nezaručuje automatický úspech a výhru bitky.

Posledný väčší rozdiel je v skombinovaní ťahovej časti s real-time bitkami. Hra tak oddeľuje plánovanie postupu a vylepšovanie vlastných miest od bitiek. Všetky hry popísané v predchádzajúcej kapitole obsahujú iba jeden z týchto prvkov, ale nie ich kombináciu. Hry obsahujú buď real-time bitky v ktorých hráč stavia a bojuje zároveň, alebo iba ťahovú mapu v ktorej tieto bitky chýbajú alebo sú nahradené napríklad kartovou hrou.

3 Dostupné technológie

Popis niektorých frameworkov vhodných na tvorbu hier s 2D alebo izometrickou grafikou.

3.1 AndEngine

Tento Java framework vytvoril Nicholas Gramlich pre tvorbu 2D hier pre OS Android vzužívajúcich technológiu OpenGLES 2.0. Tento engine je jednoduchý na zorientovanie sa v ňom, má mnoho doplnkov a možností medzi ktoré patrí napríklad aj implementovaná fyzika. Jeho veľkou nevýhodou sú neaktualizované knižnice ktoré napríklad nespolupracujú s Java updatom 1.7. V súčastnosti sú dve verzie tohto enginu [3, 4] :

- GLES 1.0
- GLES 2.0

Jednou zo základných komponent enginu je trieda `BaseGameActivity` obsahujúca štyri hlavné metódy:

- onLoadEngine
- onLoadResources
- onLoadScene
- onLoadComplete

Tieto metódy postupne načítavajú engine, zdroje, hernú logiku a vytvárajú hotovú scénu o čom vypovedajú aj ich samotné názvy [5].

Engine sa stal v poslednej dobe veľmi populárnym pre vývojárov 2D hier a už teraz je na trhu mnoho hier vytvorených vďaka jeho použitiu. Medzi njaznámejšie a najsťahovanejšie patria napríklad:

- Bunny Shooter
- Fireworks
- Greedy Spiders

3.2 Cocos2d-X

Multiplatformový open-source framework vyvinutý pre tvorbu 2D hier. Pôvodne to bol engine pre vyvoj hier pre operaný systém iOS pod názvom Cocos2D iPhone a používal programovací jazyk `Objective-C`. Neskôr bol upravený kôli rozšíreniu sa na platformu Android pričom stále používa rovnaké API. Obsahuje množstvo 2D komponent vďaka ktorým je možné vytvoriť aj náročnejšie 2D hry s rôznymi prvkami.

Pre vývoj hier na platforme Android tento framework používa nástroje NDK ktoré umožňujú písať aplikácie v jazyku `C++` kôli rýchlosti a plynulosti. Má vstavanú väzbu na jazyky Lua a JavaScript čo umožňuje tvorbu skriptov alebo vysoko úrovňových častí hry pričom nízko úrovňové časti projektu ostávajú vytvorené jazykom C++. Framework má dobrú podporu zo strany jeho vývojárov a komunity spolu s jednoduchými návodmi a postupami na jeho oficiálných stránkach [6].

3.3 Box2D

Je open-source framework napísaný v jayzku C++ pre simuláciu objektov v 2D prostredí. Vyvinul ho Erin Catto. Framework bol portovaný na mnoho programovacích jazykov ako napríklad `Java`, `Adobe Flash`, `Lua` alebo `JavaScript`. Jednou z hlavných častí enginu je jeho prepracovaná fyzika a preto je vhodný na tvorbu 2D hier kde je fyzika dôležitou súčasťou hry. Vyvoj pomocou tohto frameworku prebieha v programovacom jazyku `C++` vďaka čomu patrí medzi náročnejšie a nevhodné pre začiatočníkov [7].

Box2D sa skladá z troch hlavných modulov:

- Common
- Collision
- Dynamics

Modul `Common` má na starosti alokáciu zdrojov, matematiku použitú v simulácií a ostatné nastavenia. `Collision`, ako napovedá jeho názov sa stará o kolízne funkcie alebo tvary objektov. Posledný z modulov - `Dynamics` poskytuje už samotnú simuláciu pohybu objektov v 2D svete [8].

4 Implementácia hry

4.1 Štruktúra projektu

Všetky triedy [1] nachádzajúce sa v projekte hry sú rozdelené do samostatných balíčkov [2] na základe ich použitia v hre z dôvodu jednoduchšej orientácie sa v štruktúre projektu. Balíčky sú podľa týchto častí hry aj pomenované a ich rozdelenie je nasledovné:

- `gor0020.bp.map` - v balíčku sa nachádzajú triedy používané v prvej časti hry - globálnej mape. Obsahuje triedy použité pri tvorbe hlavnej konštrukcie mapy, vykresľovanie samotnej scény a aktivity ktoré ich používajú.
- `gor0020.bp.battle` - balíček obsahuje triedy použité v druhej časti hry - real time bitke a rovnako ako v predchadzájúcom balíčku obsahuje všetky triedy tvoriace jej jadro, aktivity a ďalšie dodatočné triedy.
- `gor0020.bp.sprites` - tretí balíček obsahuje triedu `Sprite` a ostatné triedy ktoré ju rozširujú a reprezentujú v hre konkrétné animované postavy vojakov v real time bitke.
- `gor0020.bp.other` - posledný balíček obsahuje iba dve triedy ktoré reprezentujú počas hry AI, hráča a ich vlastnosti.

4.2 Konštrukcia herného jadra

Obe časti hry - mapa a real-time bitka používajú rovnakú konštrukciu ktorá tvorí jadro. Táto konštrukcia je tvorená troma základnými prvkami:

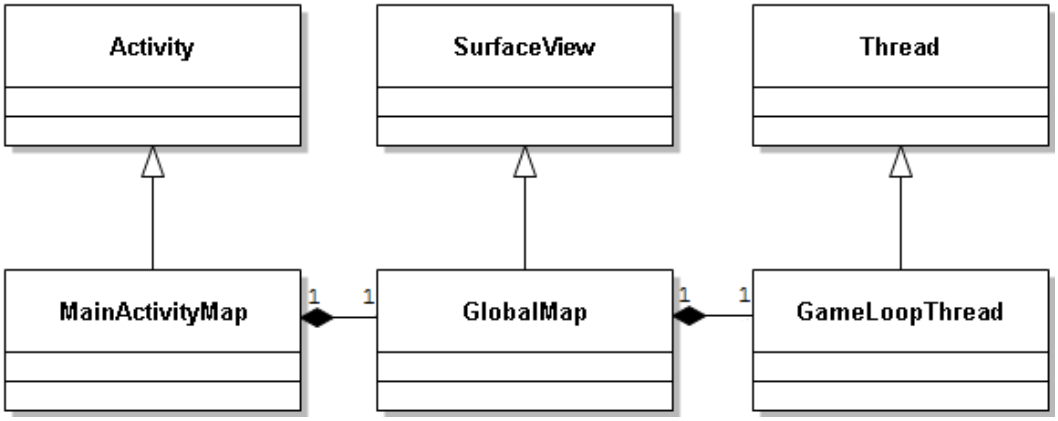
- Aktivita
- Trieda odvodená od SurfaceView
- Vykresľovacie vlákno

Tieto časti sú doplnené ďalšími triedami ktoré reprezentujú rôzne časti v užívateľskom rozhraní hry alebo zapezpečujú potrebnú funkcionalitu.

Základom konštrukcie je aktivita ktorá špecifikuje obrazovku hry a komunikuje s hráčom. Ďalšou časťou je trieda ktorá je odvodená (dedí) z triedy `SurfaceView` a reprezentuje hernú scénu. Toto riešenie umožňuje prenechať vykresľovanie paralelne bežiacemu vláknu a hlavné vlákno samotnej hry nie je zaťažované vykresľoaním ale spracováva vstupy od hráča a vykonáva ďalšie operácie. Ide o jednu zo základných možností na vykresľovanie 2D grafiky pomocou plátna [3] ktorú ponúka OS Android bez nutnosti použitia ďalšieho frameworku.

[1] všetky .class súbory
[2] anglicky package
[3] trieda Canvas

Dôvod zvolenia tejto možnosti je v jej jednoduchosti a keďže hra nie je graficky veľmi náročná pretože na vykresľovanej scéne sa objavuje menšie množstvo objektov, nemá negatívny vplyv na výkon.



Obr. 4.1: Diagram tried ukazujúci vzťah medzi triedami tvoriacimi herné jadro

Obrázok 4.1 na ktorom je jednoduchý diagram tried popisujúci vzťah medzi týmito základnými triedami tvoriacimi jadro prvej časti hry - ťahovej mapy. Rovnaká štruktúra sa nachádza aj v real-time bitke. V diagrame nie sú zobrazené premenné ani metódy tried z dôvodu jeho lepšej čitateľnosti.

4.2.1 Aktivita

Aktivita je základnou vizuálnou komponentou v Androide. Rozloženie užívateľského rozhrania aktivity ako aj hernej scény je definované v XML súbore, takzvanom Layoute. Ide o staticky vytvorené rozloženie vizuálnych prvkov. Takto vytvorené uživateľské rozhranie umožňuje oddeliť obsah od kódu. To zaisťuje lepšiu čitateľnosť samotného kódu a jednoduchú úpravu v prípade zmien. Ďalší dôvod takto vytvoreného rozhrania bol v prospech výkonu. V hre sa na spodnej lište nachádza mnoho ovládacích prvkov ako napríklad tlačítka a pri ich dynamickom vytvorení v kóde hry spolu s vykresľovaním scény pomocou `SurfaceView` môže dochádzať k zníženiu výkonu hry [9].

Aktivita reaguje na vstupy od hráča ktoré predstavujú dotyky po obrazovke. Na zachytávanie týchto vstupov definuje metódu `onTouch(View v, MotionEvent e)`. Táto metóda reaguje na tri typy dotykov:

- ACTION_ UP
- ACTION_ DOWN
- ACTION_ MOVE

Na základe týchto troch typov udalosti sú volané metódy aktivity alebo metódy nad objektom `SurfaceView` na ktorý má aktivita referenciu. Jeho metódy zabezpečujú rôzne činnosti ako napríklad posun mapy alebo označenie hradu.

4.2.2 Trieda odvodená od SurfaceView

Táto trieda dedí z triedy SurfaceView a vďaka tomu poskytuje špecializovanú plochu na ktorú je možné vykresľovať hernú scénu pomocou Canvasu a stará sa jej správne umiestnenie a zobrazenie na displeji zariadenia. Prístup k vykresľovanej ploche zaisťuje metóda `getHolder()` ktorá vracia objekt typu SurfaceView na ktorý je možno kresliť. Trieda tiež implementuje rozhranie `SurfaceHolder.Callback` ktoré umožňuje reagovať na zmenu samotného objektu pomocou trojice metód:

- `surfaceCreated()`
- `surfaceChanged()`
- `surfaceDestroyed()`

Ich samotný názov napovedá kedy sú volané. Metóda `surfaceCreated()` je volaná pri vytvorení objektu SurfaceView, vytvára a načítava všetky zdroje použité v hre počas svojho životného cyklu, napríklad textúry alebo zvuky. Pri zničení objektu sa zavolá metóda `surfaceDestroyed()` ktorá uvoľňuje všetky zdroje ktoré objekt používal aby nedošlo k zbytočnému spotrebovaniu dostupnej pamäte. Posledná, `surfaceChanged()` sa volá pri zmene objektu, napríklad pri zmene orientácie obrazovky, no táto možnosť v tejto hre nie je využitá pretože hra je celý čas v Landscape móde.

Samotné vykresľovanie scény prebieha v jednoduchom cykle v separátnom vlákne pomocou dvoch metód. Prvá z nich, `lockCanvas()` vracia objekt typu Canvas, na ktorý je možno začať kresliť a uzamyká ho pred prístupom iných objektov. Po vykreslení sa volá medóda `unlockCanvasAndPost(Canvas canvas)` ktorá berie ako parameter Canvas na ktorý bola scéna vykreslená, zobrazí ho na displeji a odomkne ho a sprístupni ďalším objektom.

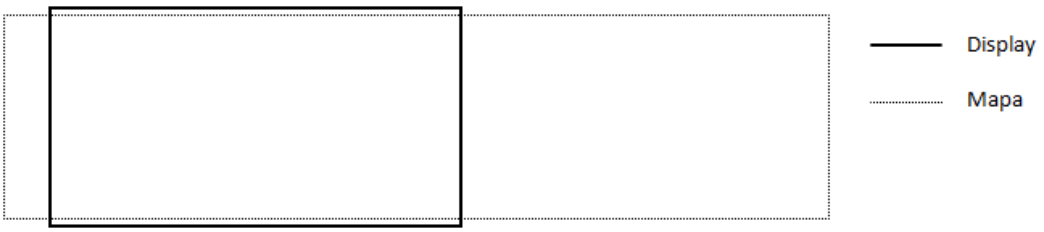
4.2.3 Vykresľovacie vlákno

Táto trieda má na starosti vykresľovací cyklus. Ide o vytvorenú vlastnú triedu ktorá dedí z triedy `Thread`, čo jej umožňuje bežať paralelne s hlavným vláknom hry. Je súčasťou objektu SurfaceView, beží v cykle a pomocou metódy `onDraw(Canvas canvas)`, ktorú volá nad týmto objektom, vykresľuje hernú scénu. Vďaka tomu Hlavné vlákno aplikácie môže vykonávať ďalšie výpočty a operácie bez toho aby bolo zaťažované kreslením scény.

4.3 Tvorba hernej mapy

Herná mapa ktorá predstavuje ťahovú časť hry je tvorená dvoma bitmapami. Jedna z nich graficky reprezentuje samotnú mapu a druhá je použitá pri rozoznávaní jednotlivých hradov ktoré sú na nej rozmiestnené a jej presnejší popis a využitie popisuje kapitola 4.3.2. Bitmapa ktorá reprezentuje graficky mapu je širšia ako displej zariadenia takže sa zobrazuje iba jej časť. Preto všetky grafické prvky nachádzajúce sa na mape, napríklad stromy, hory alebo hrady nie sú samostatné bitmapy dodatočne vykresľované ale

sú už pevnou súčasťou mapy, takže vlákno, ktoré má na starosti vykresľovanie, sa nemusí starať o zbytočné množstvo objektov. Toto šetrí výkon a aj tvorbu kódu ktorý by musel vždy zisťovať najprv polohu objektov ktorá sa mení keďže mapa sa dá posúvať do strán. Tento posun a časť vkresľovanej mapy znázorňuje obrázok 4.2.



Obr. 4.2: Ukážka vykresľovania časti mapy na displej

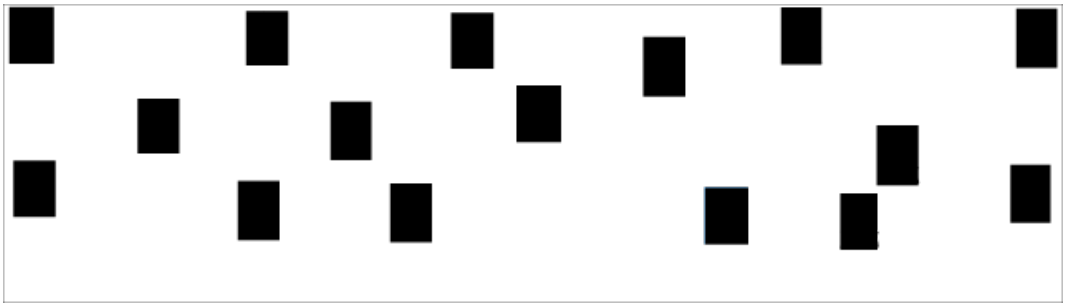
4.3.1 Posun mapy

Keďže je mapa širšia ak odisplej zariadenia, zobrazuje sa v aktuálnom čase iba jej časť. To znamená že mapa je posúvateľná zľava doprava a naopak po horizontálnej ose X. Časť ktorá má byť zobrazená je daná súradnicami ktoré orčujú počiatok a koniec vykresľovania. Ďalšiu časť možno vykresliť zmenou súradníc vykresľovania. Zmena nastáva v prípade udalosti vyvolanej dotykom a následným posunom mapy hráčom. Túto udalosť zachytáva metóda `onTouch(View v, MotionEvent e)` ktorá už bola spomínaná v kapitole 4.2.1 a vzhľadom na typ vyvolanej udalosti volá kontkrétné metódy ktorých funkcionalitu rieši vlastná implementácia. Pri dotyku a následnom posune po displeji sa najprv zachytí udalosť typu `ACTION_ DOWN`, čo predstavuje prvý dotyk a je zavolaná vlastná metóda `touchDown(Event e, Layout l)`. Táto uloží súradnice dotyku na ose X a ose Y do globálnej premennej. Pokiaľ začne hráč posúvať prst po displeji, je už zachytená udalosť `ACTION_ MOVE` ktorá volá metódu `touchMove(MotionEvent e)`. V tejto metóde sa znova získajú súradnice dotyku. Dôležitá je súradnica na ose X. Keďže bol zaznamenaný pohyb, táto súradnica je rozdielná oproti získanej pri prvom dotyku. Rozdiel medzi týmito dvoma súradnicami predstavuje počet pixelov o koľko sa má bitmapa posunúť a smer posunu. O tento rozdiel sa zmení hodnota súradnice na ose X ktorá reprezentuje počiatočný bod vykresľovania mapy. Po tejto zmene sa aktualizuje premenná obsahujúca hodnotu súradnice prvého dotyku ktorá bola získaná na začiatku v metóde `onTouch(View v, MotionEvent e)`. Týmto spôsobom sa menia súradnice kreslenia mapy pokiaľ hráč neprestane mapu posúvať alebo nedôjde k jej okraju.

4.3.2 Rozlišovanie hradov na mape

Na mape sa nachádza šestnásť hradov a každý z nich je reprezentovaný samostatným objektom vytvorenej triedy `Castle` a je uložený v kolekcii. Pri dotyku na displej je potrebné zistiť či sa hráč dotkol miesta na ktorom sa jeden z týchto hradov nachádza a ak

áno, o ktorý presne ide. K tomu slúži druhá bitmapa ktorá je veľkosťou rozlíšenia identická s vykresľovanou bitmapou graficky reprezentujúcou mapu sveta. Táto bitmapa je ale celkom priehľadná a nenachádzajú sa na nej žiadné grafické prvky okrem malých štvorcov ktorých rozloženie približuje obrázok 4.3, na ktorom sú tieto štvorce kôli viditeľnosti vyplnené čiernou farbou.



Obr. 4.3: Ukážka rozmiestnenia hradov na pomocnej bitmape

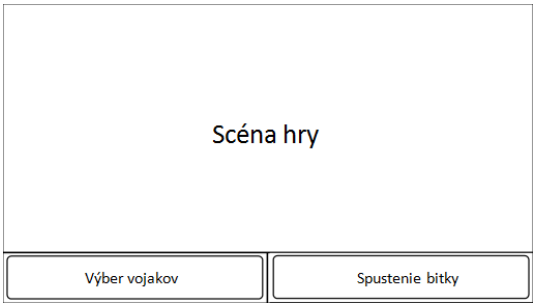
Tieto štvorce sú na rovnakom mieste ako hrady na vykresľovanej bitmape. Každý takýto štvorec je vyplnený odtieňom bielej farby s nepatrným rozdielom hodnôt v kóde ARGB stupnice. Tieto kódy farieb sú prevedené zo šestnástkovej sústavy do desiatkovej a sú uložené v premenných typu `integer` v rozmedzí -1 až -16. Napríklad hodnota `-1` reprezentuje v ARGB stupnici hodnotu `0xFFFFFFFF`. Každý z hradov má jednu túto hodnotu uloženú vo vlastnej premennej. Takto je zaručená jedinečnosť každého z hradov a je možné ich ľahko rozlíšiť.

Samotné rozlišovanie hradov začína pri zistení súradníc dotyku. Tieto súradnice zisťuje metóda `touchDown(Event e, Layout l)` ktorá je vyvolaná udalosťou pri dotyku na displej. Táto metóda už bola popisovaná v kapitole 4.3.1 popisujúcej posun hernej mapy. Po dotyku je potrebné zistiť či sa na danom mieste pomocnej bitmapy nachádza jeden zo štvorcov predstavujúcich hrady. To zisťuje metóda `getPixel(int x, int y)` ktorá pomocou súradníc osy X a Y ktoré berie ako parametre zistí kód farby nachádzajúcej sa na ich mieste. Metóda vracia hodnotu typu `integer`, ktorá predstavuje jednú z hodnôt `ARGB` kódu farby. Ak sa táto hodnota rovná nule, hráč sa dotkol miesta na ktorom sa žiaden hrad nenachádza. Pokiaľ je hodnota rozdielná od nuly, hráč sa dotkol jedného z hradov a ďalšou úlohou je zistiť ktorý to je. To je vyriešené jednoduchým cyklom ktorý prechádza kolekciu v ktorej sú uložené všetky objekty reprezentujúce hrady. Ak nastane zhoda kódu farby získaného z bitmapy s kódom jedného z hradov, ktorý má každý jedinečný, cyklus sa ukončí. Tento konkrétny hrad sa zvýrazni na mape a v spodnej lište hry kde sa nachádza užívateľské rozhranie hry sa zobrazia jeho vlastnosti.

4.4 Tvorba real-time bitky

Tvorba real-time bitky v hre je odlišná od tvorby globálnej mapy. Hlavný rozdiel je interakcia s hráčom. V bitke hra nereaguje na hráčové vstupy v podobe dotykov na animovanú plochu displeja ako je tomu v prípade mapy. V bitke hráč používa na ovládanie iba

dvojicu tlačidiel ktoré sa nachádzajú na spodnej ovládacej lište a scéna hry vykresľovaná pomocou paralelného vlákna na objekt SurfaceView iba zobrazuje bitku a animuje postavy vojakov, ktorí sa v nej nachádzajú. Vykresľovaná scéna - bojisko, je rozdelená na tri vodorovné línie zoradené pod sebou. Animované postavy vojakov sa v bitke pohybujú proti sebe po týchto líniach ktoré sú popisované v kapitole 2.2. Rozloženie obrazovky je vidieť na obrázku 4.4



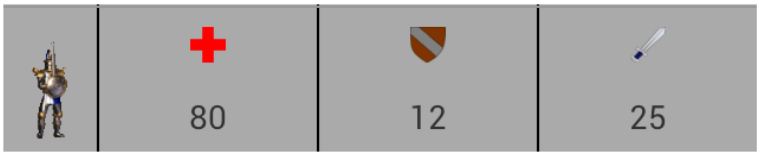
Obr. 4.4: Ukážka rozloženia prvkov na displeji v real-time bitke

Hlavná aktivita real-time bitky imlementuje vlastnú vnorenú triedu `ControlThread` ktorá je odvodená od abstraktnej triedy `AsyncTask`. Pri začatí bitky je ihneď vytvorený objekt tejto vnorenej triedy ktorý má za úlohu kontrolovať kedy nastal koniec bitky a informovať o tom hráča o jej ukončení.

Tlačidla nachádzajúce sa na spodnej časti obrazovky majú na starosti dve funkcie - voľba vojakov do bitky a spustenie samotnej bitky.

4.4.1 Pridávanie vojakov do bitky

Pri prvom spustení bitky sa na displej zariadenia raz vykreslí scéna bitky - pozadie bojiska a ďalej sa vykresľovanie nevykonáva pretože na bojisku sa ešte nenachádzajú žiadné objekty vojakov ktoré by bolo treba vykresľovať. Na pridanie objektov ktoré reprezentujú vojakov v bitke slúži tlačítko nachádzajúce sa na ľavej polovici ovládacieho užívateľského rozhrania. Stlačením tohto tlačítka sa pozastaví aktuálna aktivita, ktorá predstavuje samotnú bitku a je spustená aktivita pomenovaná `BattleActivity`. Táto aktivita zobrazuje zoznam dostupných vojakov spolu s detailom ich vlastností. Zoznam je vytvorený pomocou vlastnej triedy, ktorá je odvodená od triedy `ArrayAdapter`, čo umožnilo vytvorenie a nadefinovanie vlastného formátu riadkov použitého zoznamu ktorého ukážku možno vidieť na obrázku 4.5.



Obr. 4.5: Ukážka vlastného vzhľadu použitého v riadku zoznamu

Každá z postav vojakov je reprezentovaná objektom jednej zo štvorice tried ktoré sú odvodené od základnej triedy `Sprite`. Tieto triedy popisuje kapitola 4.5. Predtým ako sa v ľavej časti zobrazí zoznam, je potrebné skontrolovať kolekciu v ktorej sú objekty reprezentujúce vojakov uložené pretože je možnosť že táto aktivita v aktuálnej bitke už bola spustená a hráč niektorých vojakov do bitky vybral. Nedostupní vojaci sa v zozname zvýraznia tmavým pozadím aby ich hráč mohol rozoznať. Zároveň sa kontroluje aj stav jednotlivých línií a ak sa na nich nachádzajú objekty vojakov ktorí patria hráčoví a ich úroveň života klesla na nulu, sú z línie odstranení a stav línie sa zmení na neobsadený.

Pravá časť obrazovky graficky prezentuje hráčovi stav jednotlivých línií. Pri každej z týchto línií sa nachádza značka ktorá udáva ich stav - prázdná alebo obsadená. Tento stav je kontrolovaný jednoduchou podmienkou pri štarte aktivity. Pre návrat do aktivity bitky slúži talčítko na spodnej strane obrazovky, ktoré pred samotným ukončením aktuálne spustenej aktivity prevedie kontrolu stavu hráčových vojakov a línií. Pokiaľ má hráč dostatok vojakov, neobsadil každú líniu a pokúsi sa vrátiť do bitky, nie je mu to umožnené a je informovaný aby obsadil každú líniu v podobe jednoduchého informačného okna, ktoré je zobrazené pár sekúnd. Toto okno je reprezentované triedou `Toast` a je v Android aplikáciách bežne používané. Ak hráč dosadil na každú líniu vojaka, prípadne ich nemá dostatok a nemôže obsadiť každú líniu, pri stlačení tlačítka štart hra povolí návrat do aktivity reprezentujúcej bitku.

4.4.1.1 Priebeh kontroly stavu vojakov a línií

Táto kontrola prebihea pred návratom do aktivity bitky a zisťuje stav hráčových vojakov a jednotlivých línií na ktoré sú objekty reprezentujúce postavy vojakov priradené. Táto kontrola je tvorená dvoma metódami ktoré vracajú jednoduchý tátový typ `boolean`. Prvá z nich - `isPlayerEmpty()` prechádza v cykle každý objekt hráčových vojakov v kolekcii a na základe ich stavu vracia hodnotu `true` ak má hráč k dispozícií ešte nejaké jednotky, alebo vracia hodnotu `false` ak už všetkých vyčerpal. Druhá metóda - `isLineEmpty` zisťuje stav jednotlivých línií. Vracia hodnotu `true` ak hráč neobsadil niektorú z línií alebo hodnotu `false` ak hráč obsadil všetky línie a žiadna neostala prázdná.

Ak nastane kombinácia návratových hodnôt kedy metóda `isPlayerEmpty()` vráti hodnotu false a metóda `isLineEmpty()` vráti hodnotu true, znamená to, že hráč neobsadil každú líniu, pričom má k dispozícií vojakov. V tomto prípade je hráčovi zobrazená informácia aby obsadil každú líniu a hra sa nevráti do aktivity bitky. V každom inom prípade kombinácií návratových hodnôt metód sa hra vracia do real-time bitky ktorá môže byť následne spustená.

4.4.2 Spustenie bitky

Ak hráč vybral vojakov a každého priradil na jednú líniu, môže byť spustená bitka. Pred jej samotným spustením je potrebné ešte každému hráčovmu vojakovi priradiť jedného vojaka ktorý predstavuje nepriateľskú stranu. Toto priradenie je vykonané v cykle ktorý prechádza kolekciu v ktorej sú uložené objekty vojakov. V tejto kolekcii sa nachádzajú iba objekty vojakov ktorých hráč vybral a poslal do bitky, vďaka čomu cyklus zbytočne neprechádza objekty ktoré nemusí. V každej iterácii cyklu sú volané dve dôležité metódy -

`selectEnemy(Sprite playerSprite)` a `setLinesEnemy(Sprite pS)`. Obe metódy príjmajú ako parameter iterovaný objekt hráčovho vojaka.

Prvá z metód - `selectEnemy(Sprite pS)` má za úlohu vybrať vhodného nepriateľského vojaka na základe objektu hráčovho vojaka na ktorého odkaz príjma ako parameter. Najskôr v cykle prejde kolekciu, v ktorej sú uložené objekty vojakov AI. V podmienke zisťuje či je objekt nažive, nie je zároveň už v bitke a jeho línia na ktorú bol priradený sa zhoduje s líniou hráčovho vojaka. Ak sú tieto podmienky splnené, znamená to že AI už v bitke má vojaka z predchádzajúceho kola ktorý môže byť ešte použitý a nie je potrebné vyberať nového vojaka z kolekcie. V rámci splnenia tejto podmienky je zavolaná metóda `finishSelectedSoldier(Sprite pS, Sprite eS)` ktorá príjma v parametroch odkaz na hráčovho vojaka a vojaka AI. Metóda im nastaví niektoré potrebné parametre a každému z nich nastaví vzájomne jeho nepriateľa. Po dokončení tejto metódy je ukončená aj metóda `selectEnemy(Sprite pS)` pretože nie je potrebné vykonávať jej ďalšie kroky. V prípade že nenastane v podmienke počas kontrolného cyklu zhoda, AI nemá na bojisku žiadného vojaka ktorý sa nachádza na rovnakej línii ako hráčov vojak. V tomto prípade po vykonaní cyklu je volaná metóda `getBallancedType(Sprite pS)` ktorá vracia hodnotu boolean ktorá je uložená do premennej. Táto metóda konkrétne vyberá z kolekcie objekt nepriateľského vojaka na základe vlastností hráčovho vojaka, ktorého odkaz príjma ako parameter. Ak AI nemá v kolekcii vhodného vojaka ktorého môže postaviť proti hráčovmu, vyberá náhodného a metóda vracia hodnotu `true` ktorá oznamuje že bol vybraný vojak. Ak AI nemá v kolekcii už žiadného voľného vojaka ktorého by dosadila proti hráčovmu, metóda vracia hodnotu `false`. Táto metóda je podrobne popísaná v kapitole 5.1.

V ďalšom kroku metóda `setLinesPlayer(boolean find, Sprite pS)` príjma túto získanú `boolean` hodnotu ako parameter spolu s odkazom na aktuálny objekt hráčovho vojaka. Metóda na základe hodnoty `true` alebo `false` parametra `find` vyhodnotí, či AI vybrala vojaka alebo nie. Ak má parameter hodnotu `false`, AI nemá k dispozícií vojaka ktorého by postavila na líniu proti hráčovmu a tým sa stáva línia obsadená hráčom a už nie je možné na ňu dosadiť ďalších vojakov. Hodnota `true` znamená opak, AI má vojaka ktorého môže postaviť proti hráčovmu a línia nie je obsadená ani jedným protivníkom a je možné na ňu pridať vojakov. Telo metódy `selectEnemy(Sprite pS)` pre lepšiu predstavu možno vidieť vo výpise 1

```
public void selectEnemy(Sprite pS){
    for(Sprite eS : enemySprites){
        if(eS.isAlive() && eS.inBattle && eS.getLife()>0 && eS.getLine() == pS.getLine()){
            finishSelectedSoldier(eS, pS);
            return;
        }
    }
    boolean find = getBallancedType(pS);
    setLinesPlayer(find,pS);
}
```

Výpis 1: Telo metódy ktorá priradzuje hráčovým vojakom protivníkových vojakov

Týmto spôsobom je každému z hráčových vojakov priradený protivníkov vojak a zároveň sa zistí, ak AI nemá dostatok vojakov aby obsadila každú líniu.

Ďalšou možnosťou je nedostatok vojakov na obsadenie všetkých línií na strane hráča. Preto je v ďalšom kroku po metóde `selectEnemy(Sprite pS)` volaná druhá spomínaná - `setLinesEnemy(Sprite pS)`. V parametri príjma metóda odkaz na objekt hráčovho vojaka a získava líniu na ktorej sa tento vojak nachádza. Táto línia sa označí stavom ktorý o nej vypovedá že sa na nej nachádza hráčov vojak. Takto sa označí každá línia na ktorej má hráč postaveného vojaka. Ak na niektorú z línií hráč vojaka nedosadil z dôvodu ich nedostatku, tejto línii sa nezmenil stav a ostáva ako neobsadená hráčom. Z toho vyplýva, že hráč nemal dostatok vojakov na to aby líniu obsadil a táto línia sa dostáva pod kontrolu AI a už na ňu nie je možné pridávať vojakov.

Po tejto kontrole má každý hráčov vojak priradeného svojho nepriateľského vojaka alebo sa neobsadené línie označia vlajkami ich víťazov a hra sa môže vrátiť do aktivity bitky kde sa spustí paralelné vlákno ktoré má na starosti vykresľovací cyklus bitky.

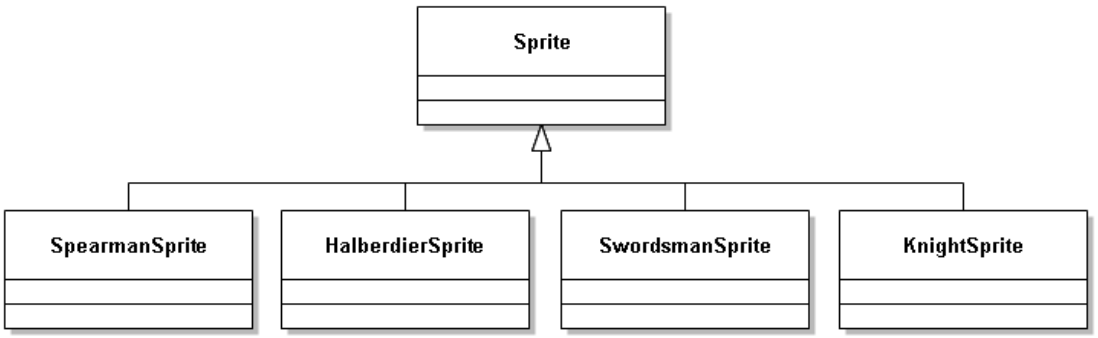
4.5 Tvorba postáv vojakov

V druhej časti hry - bitke, sa nachádzajú postavy vojakov ktoré sa pohybujú po scéne s bojujú spolu. Každý takýto vojak je reprezentovaný objektom uloženým v kolekcii s ktorou sa pracuje v priebehu bitky.

V hre sa nachádzajú štyri druhy vojakov, preto boli vytvorené štyri špecifické triedy ktoré ich reprezentujú:

- SpearmanSprite
- HalberdierSprite
- SwordsmanSprite
- KnightSprite

Všetky tieto triedy sú spoločne odvodené od jednej základnej triedy - `Sprite`. Ich hierarchiu možno vidieť na obrázku 4.6



Obr. 4.6: Hierarchia tried reprezentujúcich animované postavy vojakov

Toto rozdelenie tried umožňuje udržiavať prieh'adnejší kód a jednoducho vytvoriť aj objekt konkrétného vojaka. Pri jeho vytvoreni je najprv volaný konštruktor predka, teda konštruktor triedy `Sprite`, ktorý vytvorí premenné a vlastnosti ktoré majú všetky objekty spoločné a nastaví im požadované hodnoty. Až po jeho dokončení sa vytvoria d'alšie premenné a nastavia sa im hodnoty ktoré majú postavy vojakov rozdielne, ako napríklad bitmapa ktorá graficky reprezentuje postavu v hernej scéne. Toto všetko sa vykonáva v konštruktore konkrétneho objektu vojaka pri jeho vytváraní a nie je už potrebné objektu nastavovať d'alšie vlastnosti pomocou metód. Ak by boli totiž všetky postavy vojakov reprezentované iba jedinou triedou `Sprite`, pri vytvorení objektu by bolo nutné každému nastaviť mnoho vlastností ktorými sa líši od ostatných, čo by malo za následok neprehľadný a zbytočne dlhý kód.

4.5.1 Animácia postáv vojakov

Hlavnou a základnou súčasťou každého objektu vojaka je bitmapa ktorá ho graficky reprezentuje na hernej scéne. Každá táto bitmapa je rozdelená na tri sekvencie podľa druhu pohybu postavy vojaka - chôdza, boj alebo pád na zem. Každá táto sekvencia je zložená zo série snímok ktoré sú od seba rozdielne a zachytávajú inú polohu pohybu. Na obrázku 4.7 je vidieť sériu snímok ktoré tvoria sekvenciu chôdze vojaka po hernej scéne.

Obr. 4.7: Hierarchia tried reprezentujúcich animované postavy vojakov

Samotná animácia týchto postáv prebieha vo vykresľovacom cykle kde sa počas jednej iterácie prejde každý objekt v kolekcii. Pri každej iterácii sa na hernej scéne vykreslí jedna snímka každého objektu na určitý čas. Pri d'alšej iterácii sa znova prejde kolekcia kde sú objekty uložené a aktualizuje sa zobrazovaná snímka. Vykreslí sa teda d'alšia v poradí na určitý čas. Vykresľovanie a zmena snímkov objektu sa stále opakuje pokiaľ sa objekt nachádza na hernej scéne a má byť vykresľovaný. Takto vzniká plynulá animácia každej postavy na hernej scéne a vytvára dojem jej pohybu.

4.5.2 Detekcia kolízie objektov

V každej iterácii cyklu ktorý prechádza kolekciu s objektami vojakov je nad každým týmto objektom volaná metóda `update()`. Táto metóda kontroluje stav vojaka, zisťuje či je objekt nažive alebo ktorú zo sekvencii snímok je potrebné vykresľovať na plátno. Každý objekt vojaka uchováva referenciu na objekt vojaka svojho nepriateľa, s ktorým sa nachádza na rovnakej línií. Preto sa v metóde zisťuje aj ich poloha na ose X a vzájomná vzdialenosť medzi nimi. Ak sa hodnota vzdialenosti dostane pod určitú úroveň, je de-

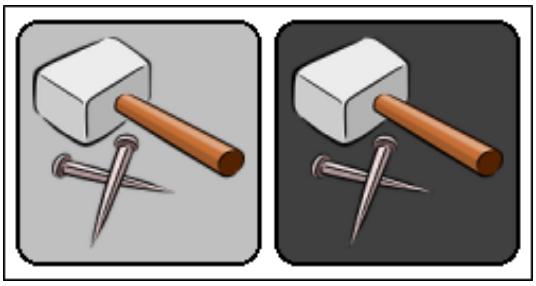
tekovaná ich vzájomná kolízia. Pri detekcii kolízie sa objekty na scéne prestanú posúvať vpred a rovnako je zvolená sekvencia snímok ktoré sú vykresľované pre boj.

4.6 Tvorba vlastných grafických komponent

Hra obsahuje mnoho grafických funkčných ovládacích a informačných prvkov. Ide hlavne o tlačítka pomocou ktorých je ovládaná a krátke správy zobrazujúce dôležité informácie a rady hráčovi na určitú dobu. Každá táto správa sa v androide nazýva `Toast`. Problémom pri týchto prvkov je ich vzhľad ktorý štýlovo a graficky nezapadá do hrý a pôsobí rušivo. Tento problém bol vyriešený návrhom a vytvorením vlastného vzhľadu týchto prvkov.

4.6.1 Tlačítka

Všetky tlačítka v hre sú vytvorené staticky v XML súboroch. Každé toto tlačítko je tvorené elementom typu `ImageButton` ktorý obsahuje atribút `android:background`. Tento atribút odkazuje na súbor uložený v zložke `drawable` v projekte kde sú uložené všetky bitmapy použité v hre. Odkazovaný súbor však nie je bitmapa, ale ďalší XML súbor obsahujúci root element typu `selector`. Tento element obsahuje niekoľko vnorených elementov typu `item` ktoré už majú v atribútoch odkaz na konkrétnu bitmapu. Každý item však obsahuje ešte niekoľko ďalších atribútov rozlišujúci stav jeho `View` objektu ktorý ho vlastní, v tomto prípade je to `ImageButton` objekt. Na základe stavu objektu, napríklad ak je tlačitko stlačené, Android vyberie vhodný `item` a jeho bitmapa je vykreslená ako pozadie celého tlačítka. Na obrázku 4.8 je zobrazené porovnanie stlačeného tlačítka a tlačítka v stave nečinnosti.



Obr. 4.8: Porovnanie tlačítka v stave stlačenia a nečinnosti

4.6.2 Toast správy

Tieto prvky sa v Androide používajú na zobrazovanie krátkých správ ktoré informujú používateľa aplikácie o nejakej zmene alebo udalosti. `Toast` správa je zobrazená v tmavošedom obdĺžniku bielym textom po určitú dobu. Doba trvania zobrazenia sa nastavuje pri vytvorení objektu `Toast` ako parameter metódy `makeText(...)` pomocou konštánt `LENGTH_SHORT` alebo `LENGTH_LONG`. Pri bežnom použití stačí vytvoriť takýto objekt s

vhodnými parametrami a zavolať nad ním metódu `show()`.V prípade tejto hry však implicitný vzhľad nie je vhodný a je potrebné ho zmeniť. Objekt je teda najprv vytvorený bez parametrov zadavaných v konštruktore ktoré sú mu postupne nastavené. Najprv je potrené vytvoriť `View` objekt reprezentujúci jeho grafický vzhľad. Tento view je vytvorený pomocou objektu `LayoutInflater` na základe pripraveného XML layoutu. Keď je view vytvorený, je mu pomocou metódy `setText(String s)` nastavený požadovaný text ktorý sa má hráčovi zobraziť. Na záver je takto vytvorený view pridá do na začiatku vytvoreného `Toast` objektu ako hlavný view. Takto vytvorenému toastu sa ešte nastavia potrebné parametre ako napríklad dĺžka jeho zobrazenia a následne je zobrazený na displej zariadenia. Na obrázku 4.9 je porovnaný implicitný systémový a vlastný toast.

Obr. 4.9: Porovnanie vzhľadu implicitného a vlastného objektu Toast

4.7 Ukladanie a načítanie hry

Dôležitou súčasťou hry je možnosť jej uloženia a opätovného načítania a pokračovania od uloženej pozície. Hra je ukladaná automaticky pri jej ukončení a navrátení do hlavného menu. V tomto menu má hráč možnosť v hre pokračovať od tejto uloženej pozície. Pre spracovanie a uloženie potrebných dát bol zvolený formát `Xml`.

4.7.1 Použitie XML

XML - eXtensible Markup Language, je značkovací jazyk ktorý umožňuje tvorbu vlastných štrukturovaných dat alebo značkovacíchj jazykoch a vyhovuje `UNICODE` štandardu. Vzhľadovo a štruktúrou pripomína jazyk `HTML`. Rozdielom je však skutočnosť že presne nešpecifikuje význam jednotlivých elementov a nemá predurčený ich obmedzený počet, nakoľko je možné ich vytvoriť veľké množstvo podľa potreby tvorcu [10, 11].

Vďaka týmto jeho výhodam bolo možné jednoducho navrhnúť a vytvoriť presnú štruktúru ukládaných dát. Ukladané sú jednotlivé objekty hradov nachádzajúce sa na mape hry a ich všetky hodnoty ich premenných. Ďalšou výhodou použitia tohto formátu bola možnosť súbor s uloženými dátami počas tvorby hry otvoriť v textovom editore, vďaka čomu sa dali jednoducho ladiť chyby algoritmu ktorý tieto dáta zapisoval.

Problémom uloženého súboru bolo jeho umiestnenie. Ak by bol súbor umiestnený v externom úložisku, čiže SD karte zariadenia, mohla by nastať situácia kedy by sa k súboru dostal užívateľ alebo nejaká iná aplikácia a jeho obsah by bol zmenený. V tomto prípade by načítavanie hry neprebiehalo korektne a bola by možnosť pádu celej aplikácie, prípadne nesprávnému načítaniu hry. Z tohto dôvodu je súbor s dátami uložený

v internej pamäti - pamäti zariadenia kde k nemu užívateľ alebo iné aplikácie nemajú prístup. Pri odinštalovaní hry je tento súbor z pamäti vymazaný systémom [12].

4.7.2 Ukladanie hry

Hra je ukladaná pri jej ukončení do pripraveného `XML` súboru. Pred samotným uložením je hráčovi zobrazený jednoduchý dialóg ktorého potvrdením je hra uložená a následne ukončená. V opačnom prípade sa hra neuloží a je ihneď ukončená.

Ukladanie prebieha v cykle, kde sa postupne prechádza celá kolekcia, v ktorej sú uložené objekty ktoré reprezentujú hrady. Na vytvorenie elementov a ich atribútov je použitá trieda `XmlSerializer`, ktorá poskytuje metódy na ich jednoduchú tvorbu. Do vytvoreného súboru sú dáta zapisované prostredníctvom vytvoreného streamu pomocou triedy `FileOutputStream`. V každej iterácii cyklu je vytvorený element `<castle>`. Hodnoty všetkých premenných ukladaného objektu sú pridané elementu ako atribúty okrem dvoch polí ktoré reprezentujú zoznam vojakov a susedných hradov uložených ako premenné typu `integer`. Tieto polia sú uložené do vnorených elementov s názvom `<soldiers>` a `<nearbyCastles>`, ktoré obsahujú uložené hodnoty týchto polí v ďalších samostatných elementov. Vo výpise kódu 2 je znázornená štruktúra ukladaného objektu.

```
<castle>
   <soldiers>
      <soldier>1</soldier>
      <soldier>1</soldier>
      <soldier>1</soldier>
   </soldiers>
   <nearbyCastles>
      <nearbyCastle>−4</nearbyCastle>
      <nearbyCastle>−6</nearbyCastle>
      <nearbyCastle>−7</nearbyCastle>
   </nearbyCastles>
</castle>
```

Výpis 2: Štruktúra objektu Castle uloženého v Xml súbore

Tento objekt obsahuje troch vojakov typu kopjník uložených samostatne v elementoch `<soldier>` a ID troch objektov hradov s ktorými tento hrad susedí uložených tiež samostatne v elementoch `<nearbyCastle>`. Po dokončení zápisu dát do súboru a jeho uložení je stream uzatvorený a hra je ukončená a vracia sa do hlavného menu.

4.7.3 Načítanie hry

Hru je možne načítať a pokračovať od uloženej pozície po jej spustení z hlavného menu. Pri stlačení tlačítka, ktoré spúšta toto načítavanie hry sa najprv skontroluje existencia `Xml` súboru s uloženými dátami. Pokiaľ súbor neexistuje, znamená to že hra je spúšťana po prvý krát alebo nebola pri predchádzajúcom hraní uložená. V tomto prípade je hráč informovaný o tom, že hru nie je možné načítať.

Ak sa však pri kontrole zistí že súbor existuje, znamená to že hra už bola uložená a súbor obsahuje dáta ktoré možno načítať, pretože samotná existencia súboru je podmienená ukladaniu hry. To znamená že súbor je vytvorený pri prvom ukladaní hry. Následne je spustená aktivita ktorá predstavuje hernú mapu. Tejto aktivite je poslaný textový reťazec `String` prostredníctvom objektu `Intent`. Tieto objekty slúžia v Androide na preposielanie informácií medzi aktivitami. Pri štarte aktivity a jej vytváraní v metóde `onCreate(Bundle savedInstanceState)` je tento objekt skontrolovaný. Ak obsahuje `String` reťazec ktorý reprezentuje príkaz načítania uloženej hry, je spustená metóda `load()` ktorá ma na starosti načítanie dát z uloženého `Xml` súboru.

Táto metóda pomocou tried `FileInputStream` a `InputStreamReader` načítava cez vytvorený stream dáta zo súboru ktoré sú následne spracované pomocou triedy `XmlPullParser`. Parsovanie `Xml` súboru prebieha v cykle `while` pokiaľ parser nevráti udalosť typu `END_DOCUMENT`. V jednej iterácii cyklu je načítaný celý element `<castle>`, jeho atribúty, všetky podelementy `<soldiers>` a `<nearbyCastles>` a ich hodnoty sú uložené do lokálných premenných. V ďalšom kroku je vytvorený objekt triedy `Castle` ktorý reprezentuje hrad na mape hry a pomocou jeho metód sú mu tieto premenné priradené. Nakoniec je tento objekt vložený do kolekcie ktorá obsahuje všetky tieto objekty hradov.

Takto metóda postupne načíta všetky elementy, vytvorí objekty ktorým nastaví parametre získané z týchto elementov a priradí ich do kolekcie. Po dokončení metódy je vytvorená kolekcia naplnená objektmi všetkých hradov nachádzajúcich sa na mape a spustí sa hra, ktorá pokračuje od poslednej uloženej pozície. Táto metóda sa nachádza v konštrukcii `try` a `catch` pre prípad možnosti vyvolania výnimky kôli napríklad zle uloženému alebo poškodenému súboru. V tomto prípade hra nie je načítana ale je vytvorená úplnne nová.

4.8 Textúry

Obrázky použité v celej hre ako textúry alebo súčasť ostatných grafických prvkov sa nachádzajú v adresároch s názvom `drawable` ktoré sa rozdeľujú podľa veľkosti rozlíšenia obrázkov. Všetky tieto obrázky sú typu PNG a sú voľne dostupné pre nekomerčné účely [16, 17, 18].

4.9 Zvuky

Zvuky použité v hre sa nachádzajú v štruktúre projektu v adresári `raw` a sú prehrávané počas hry pomocou triedy `MediaPlayer`. Tieto zvuky sú voľne dostupné pre nekomerčné účely [19]

5 Tvorba umelej inteligencie

Umelá inteligencia, v skratke `AI`, je súčasťou skoro každej hry a rovnako je vytvorená aj v tejto hre v jednoduchej forme. Vďaka rozdeleniu hry do dvoch častí - mapa a bitka, ktoré sa od seba veľmi líšia, bola vytvorené dvoma spôsobmi.

5.1 AI v bitke

AI je v bitke podstatne jednoduchá. Rovnako ako hráč má možnosť iba vybrať vojaka a pridať ho do línie do bitky. Celá činnosť výberu vojaka prebieha iba v jednej metóde - `getBallencedType(Sprite pS)` ktorá už bola spomínaná v kapitole 4.4.2.

Základom pre túto metódu je objekt `Sprite`, ktorého príjma v parametre a tento objekt predstavuje vojaka ktorého hráč zvolil do bitky alebo ho už v bitke má. Na začiatku metóda zistí typ vojaka podľa hodnoty jeho vlastnej verejnej premennej typu `integer` ktorá reprezentuje jeho typ. Podľa takto zisteného typu metóda v ďalšom kroku vyberie z kolekcie objektov vojakov AI jedného podľa priorít. Tieto priority sú zoradené od najvyššej po najnižšiu a pokiaľ nastane s jednou z nich zhoda, výber je ukončený a metóda vráti objekt zvoleného vojaka. AI sa snaží vybrať vojaka na rovnakej úrovni ako je hráčov, preto má najvyššiu prioritu zhoda, aby nebol zbytočne použitý vojak ktorý je drahý. Počas hry by mohla nastať situácia kedy by bolo potrebné zvoliť napríklad najsilnejšieho vojaka, no ak by bol zvolený na začiatku bitky, už by nebol k dispozícií a AI by musela zvoliť nejakého slabšieho. Ak AI nezistí zhodu so žiadným vlastným vojakom, je na rade porovnávanie vlastností. Tu sú porovnávané bonusové vlastnosti vojakov, napríklad kopijník je slabý proti ostatným pešiakom, no proti jazdcovi má veľký bonus v útoku čo je dobré využiť. Tieto vlastnosti popisuje kapitola 2.1.2 a jej podkapitoly. Ako posledná je možnosť výberu na základe náhody. Ak sa v priebehu hľadania v kolekcii nevybral žiaden vojak podľa prvých dvoch priorít, vyberie sa náhodne taký objekt, ktorý sa ešte v bitke nenachádza. Táto možnosť sa však použije až v krajnom prípade.

5.2 AI na mape

Na globálnej mape bola tvorba AI podstatne zložitejšia ako v bitke a je reprezentovaná objektom triedy `Enemy`. Dôvodom zložitosti v tejto časti hry je hlavne množstvo možností ktoré AI môže vykonať a keďže ťahy sú obmedzené, musí si vybrať len jednu z nich. Tento výber sa odvíja od jedného z dvoch základných stavov v ktorých sa môže AI nachádzať. Konkrétne ide o útok alebo obranu. Pred vykonaním vlastného ťahu AI najprv zistí množstvo svojich zdrojov - zlata, za ktoré je možné dokupovať jednotky alebo vylepšovať mesta. Medzi útočným alebo obranným správaním sa rozhoduje práve na základe množstva týchto zdrojov. Podľa jedného z týchto dvoch stavov AI rozdelí svoje zdroje ktoré minie počas ťahu.

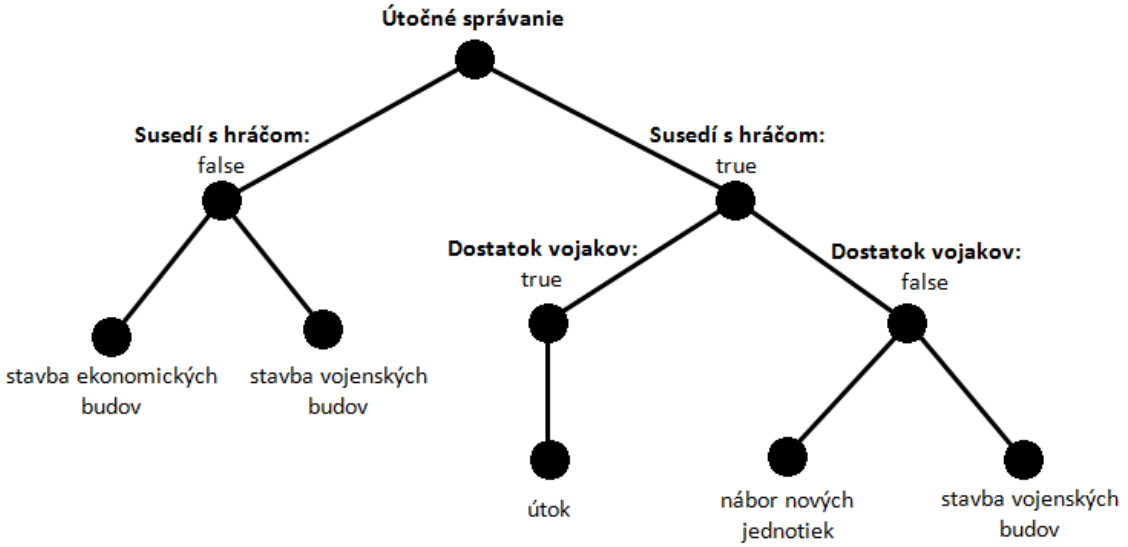
Na začiatku od celkového množstva zlata odráta určitú čiastku ktorá predstavuje žold za už zakúpené vojenské jednotky. Ak jej ostane viac ako polovica pôvodného množstva, AI má ešte dostaok financií, nastaví sa do útočného správania a zbytné zlato prerozdelí na potrebné účely. V útočnom režime vyčelní väčšiu čiastku na nábor ďalších vojakov,

prípadne na rozšírenie kasární. Ak po odrátaní žoldu za vojakov ostane menej ako polovica pôvodného množstva, AI sa nastaví do obranného správania a zbytné zlato prerozdelí inak ako v útočnom režime. Väčšiu časť financií vyčlení na vylepšenie obchodných budov ktoré zvyšujú príjmy.

V oboch týchto stavoch však AI sleduje pomer vojenskej a ekonomickej úrovne všetkých miest, aby sa zabránilo príliš veľkému rozdielu v niektorých miestach. Napríklad jedno mesto by malo príliš vysokú vojenskú úroveň, zbytočne veľa vojakov za ktorých treba veľa platiť a ekonomická úroveň je na minime. Toto porovnanie má tiež vplyv na výber útočného alebo obranného správania.

5.2.1 Útočné správanie

V tomto stave je AI agresívnejšia, väčšinu prostriedkov - zlata, vynakladá na výstavbu vojenskch budov ktoré odomykajú ďalšie vojenské jednotky, nábor vojakov alebo ak má príležitosť útočí na hráča. Na obrázku 5.1 je znázornený strom možných postupov v tomto útočnom režime.



Obr. 5.1: Strom možností AI v útočnom režime

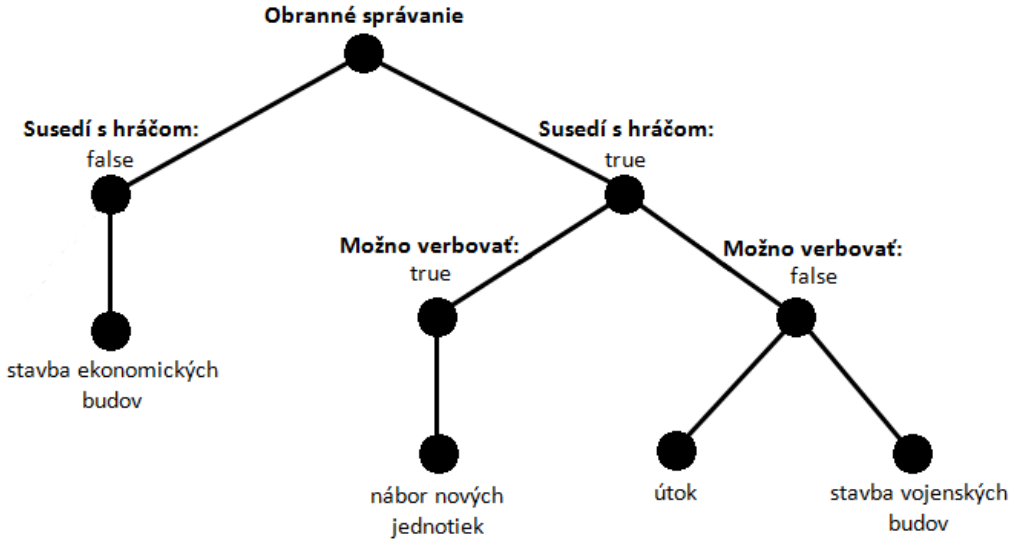
AI postupne prechádza každý svoj hrad. Najprv zisťuje prítomnosť susedného hráčovho hradu. Ak je potvrdená, v ďalšom kroku zisťuje počet svojich vojakov, ich celkovú silu a ich vzájomný pomer. Ak sú všetky tieto zistené podmienky splnené, AI má možnosť k útoku na hráčov hrad. Ak však AI nedisponuje dostatočne veľkou vojenskou silou v aktuálnom hrade, rozhodne sa pre nábor ďalších vojakov, pretože je šanca že v ďalšom ťahu hráč zaútočí. Ak nie je možné naverbovať ďalšie vojenské jednotky z dôvodu naplnenej maximálnej kapacity hradu, AI sa rozhodne pre vylepšenie vojenskej úrovne ktorá zvyšuje túto maximálnú kapacitu.

Ďalšou možnosťou je neprítomnosť hráčovho hradu. V tomto prípade sa hrad nachádza hlboko v chránenom území, z ktorého nie je možné útočiť na hráča a rovnako ani hráč nemôže zaútočiť na tento hrad. AI sa teda rozhodne medzi vylepšením vojenskej alebo ekonomickej úrovne, teda výstavbou alebo náborom ďalších vojakov. To už záleží na zistenom pomere medzi týmito úrovňami.

Aby všetky tieto možnosti neboli príliš priamočiare a postupom hrania ľahko predvítateľné, je ich súčasťou generovanie náhodného čísla, ktoré vykonanie kroku potvrdí alebo zruší. Toto číslo nemá až taký veľký vplyv na rozhodovanie, ale účelu vytvorenia dojmu nepredvídateľného správania sa AI postačuje.

5.2.2 Obranné správanie

Tento stav je presným opakom útočného správania. AI sa zameriava viac na výstavbu ekonomických budov - vylepšovanie ekonomickej úrovne na ktorú vymedzí väčšie množstvo zdrojov - zlata. Tým zvyšuje svoj pravidelný príjem na konci každého ťahu. Vylepšovanie vojenskej úrovne, nábor vojakov alebo útok majú menšiu prioritu a je na ne vymedzené menšie množstvo zdrojov. Na obrázku 5.2 je znázornený strom možných postupov v obrannom správaní.



Obr. 5.2: Strom možností AI v obrannom režime

Rovnako ako v útočnom stave, AI na začiatku zisťuje, či aktuálny hrad susedí s hráčovým. Ak áno, tak zakúpi ďalšie vojenské jednotky. Pokiaľ je naplnená kapacita, zvýši vojenskú úroveň ktorá túto kapacitu zväčšuje. Ak kapacitu vylepšením vojenskej úrovne nie je možné zvýšiť, hrad dosiahol jej maximálnú úroveň a AI sa môže rozhodnúť zaútočiť na hráča.

Ak hrad nesusedí s hráčovým, AI sa zameriava na vylepšovanie ekonomickej úrovne a tým si zaistí vyšší príjem v ďalších ťahoch.

6 Technické problémy

Počas tvorby hry som sa stretol s niekoľkými problémami. Táto kapitola približuje tie najzávažnejšie, kedy k nim dochádzalo a akým spôsobom boli vyriešené. Záverečná podkapitola približuje ladenie hry a zariadeni,a na ktorých bola hra ladená.

6.1 Spotreba pamäte

Hra používa pri svojom behu množstvo textúr ktoré sú často vykresľované v rovnakom momente na displej zariadenia. Tieto textúry sú bitmapy ktoré sú načítané zo zdrojov aplikácie. Problémom bola veľkosť týchto bitmáp, pretože niektoré majú vysoké rozlíšenie kôli výslednej kvalite. Každá android aplikácia má určitý limit pamäte, približne od 16 MB do 48 MB, ktorý závisí od typu zariadenia [13]. Pri behu hry počas načítania veľkého množstva bitmáp hra zamŕzala a neskôr úplne spadla. Dôvodom týchto pádov bolo využitie všetkej dostupnej pamäte a to nie len kôli aktuálne používaným načítaným bitmapám, ale aj uchovavaným referenciám na predchádzajúce aktivity v podobe objektov typu `Context`.

Tento problém bol vyriešený niekoľkými zmenami v aplkácii. Jednou z nich bolo neudržiavanie referencií predchádzajúcich aktivít a uvoľňovanie ich načítaných bitmáp pomocou metódy `recycle()`. Táto metóda uvoľňuje objekt ktorý je asociovaný na konkrétnú bitmapu. Metóda neuvoľňuje zdroje synchronne, objekt označí ako `dead` a garbage collector ho neskôr uvoľní z pamäte[14]. Ďalším riešením bolo upravenie elementu `<aplication>` v súbore manifest pridaním parametra `android:largeHeap="true"`. Toto nastavenie umožní používať všetkým procesom aplikácie väčšie množstvo pamäte.

6.2 Hardwarová akcelerácia

Hardwarová akcelerácia vykonáva pomocou `GPU` všetky kresliace operácie, ktoré sa používajú na vykresľovanie `View` objektov. Jej použitie má však dopad na vyššiu spotrebu pamäte `RAM`. Ďalšou nevýhodou je skutočnosť, že nepodporuje všetky 2D kresliace operácie čo rovnako ako využitie všetkej dostupnej pamäte viedlo k pádom aplikácie[15]. Kôli týmto problémom musela byť hardwarová akcelerácia úplne vypnutá pridaním parametru `android:hardwareAccelerated="false"` do elementov `<activity>` v súbore manifest.

6.3 Synchronizovaný prístup k Canvasu

Keďže hra používa na vytvorenie scény triedu `SurfaceView`, ktorá vykresľuje scénu pomocou vlastného paralelne bežiaceho vlákna, vznikal problém pri dotykoch po kreslenej ploche ktoré menili súradnice a vlastnosti kreslených objektov. Tieto dotyky viedli k pádu aplikácie alebo zlému vykresleniu scény z dôvodu nesynchronizovaného prístupu k objektu `Canvas` alebo k objektom ktoré sú naň vykresľované v priebehu kresliaceho cyklu. Preto musia všetky metódy ktoré manipulujú s vlastnosťami kreslených objektov alebo so samotným objektom `Canvas` mať celý vykonávaný kód vo vnútri bloku

`synchronize(getHolder()){}`. Tento blok špecifikuje objekt ktorý má byť uzamknutý pomocou metódy `getHolder()`. Týmto objektom je samotný `SurfaceView` ktorému je možné po jeho uzamknutí zmeniť parametre pred jeho vykreslením. Počas vykonávania kódu v tomto bloku k objektu nemá prístup žiadna iná metóda a preto nedochádza k nesynchronizovanému prístupu, ktorý môže spôsobiť pád hry.

6.4 Ladenie hry

Ladenie hry patrilo medzi obtiažnú časť a dalo by sa tiež zahrnúť medzi problémy. Týmto problémom je veľká rozdielnosť zariadení ktoré ktoré používajú systém Android, konkrétne rozdielnosť v rozlíšení displejoch a ich veľkosti. Preto bolo dôležite počas vývoja hry upravovať rozlíšenie textúr a umiestňovať ich do správných adresároch `drawable`. Ani to však nezaručilo správne rozmiestnenie niektorých komponent. Konkrétne ide o zariadenia s malým displejom kde sa prejavila chyba zobrazenia užívateľského rozhrania mapy v hre oproti zariadeniam s väčším displejom. Niektoré prvky rozhrania sa nezobrazovali korektne alebo vôbec, keďže bol displej primalý na ich zobrazenie. Tento problém bol vyriešený pridaním možnosti posúvať toto rozhranie horizontálne do strán podobne ako mapu.

Počas tvorby hry som mal na jej ladenie k dispozícií tri rôzne zariadenia: Sony Xperia S, Sony Ericsson Xperia Mini Pro a tablet Asus Nexus 7.

Sony Xperia S

Rozlíšenie: `1280x720`
Veľkosť displeja: `4.3"`
Procesor: `1.5 GHz Dual Core`
RAM: `1GB`

Sony Ericsson Xperia Mini Pro

Rozlíšenie: `480x320`
Veľkosť displeja: `3.0"`
Procesor: `1 GHz Single core`
RAM: `512 MB`

Google Nexus 7

Rozlíšenie: `1280x800`
Veľkosť displeja: `7"`
Procesor: `1.2 GHz Quad core`
RAM: `1 GB`

7 Záver

Cieľom tejto práce bolo vytvoriť strtegickú hru na platformu Android. Zadanie práce som splnil a hru vytvoril podľa návrhu. V porovnaní s hrami opísanými v kapitole 2.3 som sa snažil vytvoriť hru ktorá obsahuje možnosti ktoré by hráč od strategickej hry očakával ale nemá ich až príliš mnoho, pretože hru robia neprehľadnú a zložitú. Väčšina porovnávaných hier a tiež veľké množstvo súčastných nových hier sa odohráva v prostredí fantasy, preto som zvolil pri návrhu hry obyčajné stredoveké prostredie. Ďalším rozdielom je už rozdelenie hry na ťahy a real-time bitky, čo obsahuje málo hier s podobným žánrom pre Android. Hra tak nepôsobí stereotypne a hráč strieda prostredia ktorých štýl hrania je odlišný.

Tvorba tejto hry mi umožnila zlepšiť sa v programovaní v jazyku Java a hlavným prínosom bolo rozšírenie si vedomostí v oblasti tvorby aplikácií pre systém Android. V priebehu implementácie hry som sa stretol aj s rôznými problémami o ktorých som veľa nevedel, hlavne problém s dostupnou pamäťou. Zistil som že niektoré veci nie je ľahké vytvoriť presne podľa pôvodného návrhu bez kompromisov alebo zmien.

Hra bola ladená na zariadení Sony Xperia S a Sony Xperia Mini Pro. Vďaka tomu je hra odladená pre zariadenia s rôznou veľkosťou displeja a rozlíšenia, čo je práve pri systéme Android celkom problém vzhľadom na počet a rozdiely týchto zariadení oproti konkurenčnému systému iOS od spoločnosti Apple.

O operačný systém Android sa zaujímam aj vo svojom voľnom čase a rovnako aj tvorbe jednoduchých aplikácií zatiaľ iba pre vlastné použitie, takže v budúcnosti mám v pláne túto hru pretaviť do úplného 3D prostredia s využitím jedného z dostupných 3D frameworkov.

8 Literatúra

[1] ZECHNER, Mario. *Beginning Android games: A Brief History of Android*, New York: Apress, 2011, s. 2. ISBN 1430230428.

[2] A Brief History of Android. *TechnicaMix* [online]. 2012 [cit. 2013-03-12]. Dostupné z: http://www.technicamix.com/2012/06/21/a-brief-history-android/

[3] Getting Started with AndEngine. *AndEngine Guides* [online]. 8.9.2011 [cit. 2013-03-23]. Dostupné z: http://andengineguides.wordpress.com/2011/09/08/getting-started-with-andengine/

[4] DELFIN, Ricardo. AndEngine 1: Introduction to the AndEngine. *IBM: FRC Tamán Keet Blog* [online]. 29.1.2013 [cit. 2013-03-16]. Dostupné z: https://www.ibm.com/developerworks/mydeveloperworks/blogs/TamanKeet/entry/andengine_1_introduction_to_the_andengine_android_videogame_libraries24?lang=en

[5] Getting Started: The Work Order of AndEngine. *AndEngine Guides* [online]. 10.9.2011 [cit. 2013-03-23]. Dostupné z: http://andengineguides.wordpress.com/2011/09/10/getting-started-the-work-order-of-andengine/

[6] WEISS, Nat. Cocos2d-X: Why is it one of the best free game engines? *Paralaxer* [online]. 2012, 22.8.2012 [cit. 2013-03-16]. Dostupné z: http://paralaxer.com/why-cocos2d-x-best-game-engine/

[7] About. *Box2D: A 2D Physics Engine for Games* [online]. 2011 [cit. 2013-03-16]. Dostupné z: http://box2d.org/about/

[8] CATTO, Erin. Box2D v2.2.0 User Manual: 1.6 Modules. *Box2D: A 2D Physics Engine for Games* [online]. 2011 [cit. 2013-03-16]. Dostupné z: http://www.box2d.org/manual.html

[9] SurfaceView. *Android Developers: Reference* [online]. 2008, 21.3.2013 [cit. 2013-03-25]. Dostupné z: http://developer.android.com/reference/android/view/SurfaceView.html

[10] Úvod do XML. *Digitálné média: Výukové materiály* [online]. 2012 [cit. 2013-04-20]. Dostupné z: http://tutorial.wz.cz/data/css/AVT.02.00.01.tema.html

[11] XML v desiatich bodoch. *World Wide Web Consortium* [online]. 24.8.204 [cit. 2013-04-20]. Dostupné z: http://www.w3.org/2004/08/XML-in-10-points-sk.html

[12] Storage Options: Using the Internal Storage. *Android Develpoers: API Guides* [online]. 2008 [cit. 2013-04-20]. Dostupné z: http://developer.android.com/guide/topics/data/data-storage.html#filesInternal

[13] Displaying Bitmaps Efficiently. *Android Developers: Training* [online]. 2008 [cit. 2013-04-25]. Dostupné z: http://developer.android.com/training/displaying-bitmaps/index.html

[14] Bitmap. *Android Developers: Reference* [online]. 2008 [cit. 2013-04-25]. Dostupné z: http://developer.android.com/reference/android/graphics/Bitmap.html#recycle()

[15] Hardware Acceleration. *Android Developers: API Guides* [online]. 2008 [cit. 2013-04-25]. Dostupné z: http://developer.android.com/guide/topics/graphics/hardware-accel.html

[16] *Pixabay* [online]. 2012 [cit. 2013-04-26]. Dostupné z: http://spritedatabase.net/

[17] *Sprite Database* [online]. 2012 [cit. 2013-04-26]. Dostupné z: http://pixabay.com/

[18] *Indie Gamer Designer* [online]. 2012 [cit. 2013-04-26]. Dostupné z: http://www.indiegamedesigner.com/

[19] *freeSFX* [online]. 2012 [cit. 2013-05-2]. Dostupné z: http://www.freesfx.co.uk/

A Obsah priloženého CD

1. Text bakalárskej práce vo formáte PDF, súbor: bakalarska_ praca.pdf
2. Projekt hry v Eclipse, zložka: StrategicGame
3. Vytvorený .APK súbor hry, súbor: StrategicGame.apk